DEC STD 049-0 Document Transmission (DX) Standard

DOCUMENT IDENTIFIER: A-DS-EL00049-00-0000 Rev A1, 20-Apr-1990

ABSTRACT: This standard defines protocol levels for document transmission in the office interconnect environment. This includes:

1. a data communication physical level
2. a data link level
3. an application level

It also includes:

1. a definition of "document" for use by office products
2. a file format for use by layered office

APPLICABILITY: This standard applies to computer systems that send text to or receive text from any DEC office system products in CX, DX, or AX mode. By extension it applies to computer systems which do not communicate directly with office system products but which communicate with or use files generated by other computers systems or layered products that communicate with DEC office system products (the extended family of office system products.)

This standard does not apply to computer systems which have no requirement to communicate with DEC office system products in CX, DX, or AX modes, or the extended family of DEC office system products.

STATUS: APPROVED 20-Apr-1990; use VTX SMC for current status.

This document is confidential and proprietary, and is the property of Digital Equipment Corporation. It is an unpublished work protected under the Federal copyright laws.

©Digital Equipment Corporation. 1990. All rights reserved.

DEC STD 049-0 Document Transmission (DX) Standard

DOCUMENT IDENTIFIER: A-DS-EL00049-00-0000 Rev A1, 20-Apr-1990

REVISION HISTORY:	A version prior to Rev A, and dated 30-Nov-1983, was submitted for SARA review but was not released.		
	Rev A,	17-Jan-1986	
	Rev A1,	20-Apr-1990	ECO Number CTS01
Document Management Category:	Document Transmission (SB)		
Responsible Department:	Applications and Time Sharing Engineering (ATSE)		
Responsible Person:	Al Breveleri		
Author:	Don Sharp		

APPROVAL: This standard was initially reviewed and recommended for approval by the Engineering Committee. Rev A1 incorporates clerical changes only.

Al Breveleri - Applications and Time Sharing Engineering

Eric Williams - Standards Process Manager

Direct requests for further information to:

Al Breveleri
Use $ VTX ELF for the latest location information.

Use VTX SMC to order copies of this document from Standards and Methods Control. Send distribution questions to JOKUR::SMC or call DTN: 234-4423.

CONTENTS

1 INTRODUCTION 1
1.1 MOTIVATION 1
1.2 GOALS 1
1.3 NON-PLANNED EFFECTS OF GOALS 1
1.4 NON-GOALS 1
1.5 SCOPE 2
1.6 HISTORY OF PREVIOUS STANDARDIZATION 2
1.7 RELATED CURRENT STANDARDS 2
1.8 FUTURE STANDARDS ACTIVITIES 2
1.9 KNOWN INCOMPATIBILITIES WITH CURRENT SOFTWARE 2

2 TERMINOLOGY 3
2.1 CX 3
2.2 DX 3
2.3 AX 3
2.4 PACKETS 3
2.5 PROTOCOL 3
2.6 LOW LEVEL 3
2.7 HIGH LEVEL 3

3 CONFORMANCE 4

4 OVERVIEW OF THE STANDARD 4

5 DEFINITION OF THE STANDARD 4
5.1 PHYSICAL LAYER 4
5.2 LOW LEVEL DX 4
5.2.1 Low Level Functions 5
5.2.2 Pipeline Operation 5
5.2.3 Cancellation 5
5.2.4 Low Level Timing 6
5.2.5 Simultaneous Sending And Receiving 6
5.2.6 Low Level DX Versus Low Level AX 6
5.3 HIGH LEVEL DX 7
5.3.1 Initialization 7
5.3.2 Operation 7
5.3.2.1 Sending A Message 8
5.3.2.2 Sending a Document 8
5.3.2.3 Receiving a Document 9
5.3.2.4 Halting a Transfer 10
5.3.3 Cancellation 10
5.3.4 High Level Timing 10
5.3.5 High Level DX Versus High Level AX 10

6 VERIFICATION/ENFORCEMENT 10

Appendix A PACKETS 11
A.1 SUMMARY OF PACKET TYPES 11
A.2 PACKET FORMATS 12

Appendix B THE FORMAT OF A WPS DOCUMENT AS TRANSMITTED BY DECDX 15

Appendix C FILE FORMAT FOR LAYERED PRODUCTS 19

Appendix D FILENAME CONVENTIONS 21

Appendix E CONVERSION FROM ASCII TO WPS FORMAT 23

Appendix F REPRESENTATION OF NUMERIC VALUES 27

Appendix G PROTOCOL STATE TRANSITION DIAGRAMS 29

Appendix H DX LIBRARY 35
H.1 INTRODUCTION 35
H.1.1 APPENDIX STRUCTURE 35
H.1.2 OVERVIEW 35
H.1.3 LINKING WITH THE DX LIBRARY 35
H.1.4 CALLING LIBRARY ROUTINES 36
H.1.5 ORGANIZATION OF THE DX LIBRARY 36
H.1.5.1 DECmate-Specific Routines 36
H.1.5.2 Document-Level Routines 36
H.1.5.3 Packet-Level Routines 37
H.1.5.4 Easycom Routines 37
H.2 DX LIBRARY ROUTINES 37
H.2.1 DECMATE-SPECIFIC ROUTINES 37
H.2.1.1 SEND DECMATE DOCUMENT 37
H.2.1.2 RECEIVE DECMATE DOCUMENT 39
H.2.2 DOCUMENT-LEVEL ROUTINES 40
H.2.2.1 AX SEND DOCUMENT 40
H.2.2.2 AX RECEIVE DOCUMENT 41
H.2.2.3 DX SEND DOCUMENT 42
H.2.2.4 DX RECEIVE DOCUMENT 43
H.2.3 PACKET-LEVEL ROUTINES 45
H.2.3.1 AX CONNECT 45
H.2.3.2 DX CONNECT 46
H.2.3.3 SEND PACKET 46
H.2.3.4 RECEIVE PACKET 47
H.2.3.5 DISCONNECT 48

H.2.4 EASYCOM ROUTINES 49
H.2.4.1 CHECK IF REMOTE SUPPORTS EZCOM 49
H.2.4.2 SWITCH REMOTE TO AX MODE 49
H.2.4.3 SWITCH REMOTE TO CX MODE 50
H.2.4.4 SWITCH REMOTE TO DX MODE 50
H.2.4.5 RETURN REMOTE TO CX MODE 50
H.2.4.6 CLOSE AN EZCOM CHANNEL 51
H.3 USER-SUPPLIED ROUTINES 51
H.3.1 DOCUMENT-LEVEL INPUT/OUTPUT ROUTINES 51
H.3.1.1 Producer 51
H.3.1.2 Consumer 52
H.3.2 DOCUMENT HEADER VERIFICATION ROUTINES 53
H.3.2.1 Construct-Document-Header Routine 53
H.3.2.2 Parse-Document-Header Routine 53
H.4 CONDITION HANDLING 54
H.5 CALLABLE LIBRARY OVERVIEW 54

Appendix I REFERENCED DOCUMENTS 55
I.1 EL-Class Digital Documents 55

1 INTRODUCTION

1.1 MOTIVATION

Problems and constraints:

Installed WPS products of Digital Equipment Corporation supply a data communications capability known as CX/DX/AX.

Digital supplies mainframe and minicomputers with data communications capability.

Digital contemplates development of new products with data communications capability.

There are markets which demand compatibility of the data communications products of Digital.

Digital intends to address these markets and hopes to succeed in the engineering, marketing, manufacturing, sales, and support of products in these markets.

There are differences in the kind of data that are represented by the individual products offered by Digital (for example, formatted text on word processing systems as opposed to internally structured user-defined records in a DBMS system).

There are differences in representation of the same data among the individual products offered by Digital (for example, 7-bit characters on PDP–11s, 6-bit characters on PDP-8s).

1.2 GOALS

The goals of this standard are to:

- Completely and unambiguously specify the data communications protocol used by Digital office system products among themselves.
- Completely and unambiguously specify the data representation used by Digital office system products.
- Define in terms of other Digital, industry, national and international standards the assumptions made by Digital office system products that allow them to successfully communicate data as specified in this standard.
- Define a mapping of the data representation used by Digital office system products into an international data interchange standard.
- Define a procedure whereby an alleged implementation of the standard may be verified.

1.3 NON-PLANNED EFFECTS OF GOALS

None are identified.

1.4 NON-GOALS

It is not the goal of this standard to:

- Specify all data that will be represented by every Digital product.
- Specify a format sufficient for representation of all existing data.
- Specify a data communication protocol that is compatible with all existing Digital products.

- Implement the proposed standard for any specific Digital product.
- Define an enforcement procedure for the proposed standard.

1.5 SCOPE

This standard applies to computer systems that send text to or receive text from any Digital office system products in CX, DX, or AX mode. By extension it applies to computer systems which do not communicate directly with office system products but which communicate with or use files generated by other computers systems or layered products that communicate with Digital office system products (the extended family of office system products).

This standard does not apply to computer systems which have no requirement to communicate with Digital office system products in CX, DX or AX modes, or the extended family of Digital office system products.

1.6 HISTORY OF PREVIOUS STANDARDIZATION

There have been no published standards in this area. An implicit standard has developed inasmuch as there exist products which communicate using the DX protocol.

1.7 RELATED CURRENT STANDARDS

EL-00052-01	*DEC STD 052-1 Operational Requirements for Serial Terminals and Computer Interfaces Operating as DTEs Connected to EIA RS-232-C or CCITT V.28 Modems*
EL-00111-00	*DEC STD 111-0 DEC Standard for Terminal Synchronization*
EL-00138-00	*DEC STD 138-0 Registry of Control Functions For Character Imaging Devices*
EL-00169-00	*DEC STD 169-0 DEC Standard Coded Character Sets For Hardware and Software*

1.8 FUTURE STANDARDS ACTIVITIES

This standard addresses specific office application capabilities. As other office applications are developed, the need to extend or replace this standard will arise.

1.9 KNOWN INCOMPATIBILITIES WITH CURRENT SOFTWARE

1. DECmail V2.0 departs from this standard by including ASCII formatting characters in documents exported to VMS.
2. DECmail V2.0 departs from this standard by not including the name of the document in the header of the document
3. DX/TOPS-10 V1.0, DX/TOPS-20 V1.0 and DX/VMS V1.0, V1.1 and V1.2 depart from this standard by using byte counted records with implied FORTRAN carriage control to store documents.
4. DX/RSTS V3.1 and DX/11M V3.1 depart from this standard by including extraneous information in the document header, and by not including the name of the document in the document header.
5. DECmail V2.0, DX/TOPS-10 V1.0, DX/TOPS-20 V1.0, DX/VMS V1.0, V1.1 and V1.2, DX/RSTS V3.1 and DX/11M V3.1 depart from this standard by not using the file extension "DX" for host system files containing documents.

6. WPS-8 departs from the standard by not transmitting the language of origin code in the START OF DOCUMENT packet.

2 TERMINOLOGY

2.1 CX

CX stands for Character Transmission. CX mode conforms to the EIA RS-232-C standard for data transmission.

2.2 DX

DX stands for Document Transmission.

2.3 AX

AX stands for Automatic (document) Transmission.

2.4 PACKETS

All DX/AX information is transmitted in packets.

A DX packet is a group of ASCII characters ending with a carriage return. A DX packet has a type, a sequence number, some text and a checksum. The sequence number and checksum allow the receiver of the packet to detect transmission errors. The packet type is a character which identifies the purpose of the packet, meaning its use in the DX protocol. The text of the packet is dependent on the type.

Packets do not contain any address fields (in contrast with the packets defined by DEC STD 111-0 or CCITT recommendation X.25) and no packet switching functions are specified or implied.

2.5 PROTOCOL

The protocol is a set of rules for transmitting data. The purpose of these rules is to synchronize and control the transmission, and to detect and recover from transmission errors. The protocol specifies the order in which packets can be transmitted, and the type of data contained in the packet.

2.6 LOW LEVEL

The low level protocol is concerned with initializing and canceling a DX connection, detecting transmission errors and correcting errors by retransmission. This is analogous to the data link layer of the International Standards Organization's Basic Reference Model for Open Systems Interconnection.

2.7 HIGH LEVEL

The high level protocol is concerned with transmitting data as specified by the user. This includes sending a document, receiving a document, sending a message, receiving a message, and halting a document transfer in progress.

3 CONFORMANCE

Single- or multi-user systems which intend to communicate with the extended family of office system products in DX mode must conform to the line protocol as specified in section 5. Layered products on TOPS-10, TOPS-20, VMS, RSX- 11M, RSX–11M PLUS, and RSTS/E which intend to use files generated by office system communications products must conform to the file format standard as specified in Appendix C.

4 OVERVIEW OF THE STANDARD

The standard defines a two level line protocol for data communications. It defines user level functions possible with the protocol. It defines file formats for layered products on TOPS-10, TOPS-20, VMS, RSX–11M, RSX–11M PLUS and RSTS/E systems.

5 DEFINITION OF THE STANDARD

5.1 PHYSICAL LAYER

DX Products assume a physical layer interface as specified by *DEC STD 052-0 Operational Requirements for Serial Terminals and Computer Interfaces Operating as DTEs Connected to EIA RS-232-C or CCITT V.28 Modems*.

The data transmitted by the physical layer are 7-bit ASCII characters.

5.2 LOW LEVEL DX

At low level the DX protocol is concerned with the detection and correction of errors occurring in the physical medium. The high level supplies the low level with a packet type and packet text.

Packets are formatted as follows:

f1: Contains the packet type, if low level. If high level, contains the sequence number.

f2: Contains the sequence number if low level. If high level, contains the packet type.

Sequence numbers are in the range of 40-137 (octal). Packet types are in the range of 140-176 (octal). Note that the ranges of sequence numbers and packet types is mutually exclusive. This allows the low level to differentiate among incoming packets by type.

packet text: Up to 64 bytes in the range 40-176 (octal).

checksum: The arithmetic sum of fields f1, f2, and packet text. The checksum is a 12-bit number encoded into two characters. (If the actual sum of the characters sent is greater than 4095, then only the low 12 bits are used as the checksum). See Appendix F for an explanation of representation of transmitted numeric values.

RETURN: The carriage return, octal code 015. The presence of the RETURN defines the length of the packet.

For coded representations of packet types, see Appendix A.

At the low level, *DEC STD 111-0 DEC Standard for Terminal Synchronization* applies. To prevent characters from arriving faster than a unit can dispose of them, every unit ceases to send characters if it receives DC3 (octal 023, also known as XOFF) and resumes sending where it left off when it receives DC1 (octal 021, also known as XON).

5.2.1 Low Level Functions

An implementation of the low level of DX protocol must have the ability to:

1. Buffer packets supplied for transmission by the high level.
2. Block execution of the high level producing process if the limit of unacknowledged transmitted high level packets has been reached.
3. Allocate a number for the packet. Numbers are allocated sequentially modulo 40 octal, using the representation scheme as specified in Appendix F. This is known as the sequence number.
4. Calculate the checksum of the text characters, packet type character, and sequence number and append the checksum characters to the packet, followed by a RETURN (octal 15).
5. Transmit the packet on the physical level.
6. On receipt of ACK or NAK, update the local record of the last packet successfully transmitted.
7. Retransmit packets if necessary because of NAK or timeout.
8. Compare the received checksum and the recalculated checksum of a received high level packet, send NAK if they differ.
9. Compare the received sequence number and the expected sequence number of a received high level packet, send NAK if they differ.
10. Send ACK for the received high level packet if the sequence number and checksum are found to be correct.
11. Supply the type and text of a correctly received high level packet to the high level.

See Appendix G for a state transition diagram for the low level protocol of DX and AX. The diagrams depict protocol states in terms of what packets may be received or transmitted, and the new state resulting from each transmission or reception.

5.2.2 Pipeline Operation

The DX protocol allows pipelining of packets, meaning that it is permissible to send a packet while previous packets remain unacknowledged. As many as five MESSAGE packets may be transmitted without waiting for acknowledgment. A maximum of two TEXT packets may be transmitted without waiting for acknowledgment. At other times the high level dialog requires a high level response after only one packet.

5.2.3 Cancellation

A QUIT packet received at any time terminates the operation in progress. The DX station then enters the STOPPED state, and operator intervention is required to reinitialize. An AX station will enter the INITIALIZING state; no local operator intervention is required to reinitialize the AX station.

5.2.4 Low Level Timing

During low level operation there are two timers ticking. The 5-second timer controls retransmission of unacknowledged packets. The 50- second timer controls the change in state from operational to stopped.

The 5-second timer is set for five seconds after the transmission of each high level packet. It is reset at the start of each transmission, and with the receipt of any low level packet. If this timer expires all unacknowledged packets are retransmitted. (Only high level packets require low level acknowledgment. Low level packets do not require retransmission, so they need not be acknowledged).

The 50-second timer is set after transmission of each new high level packet. It is reset at the start of each new transmission, and with the receipt of any low level packet. This timer expires when there has been no reception from the remote station for more than 50 seconds. The low level then signals the high level that the connection has been lost.

5.2.5 Simultaneous Sending And Receiving

The DX protocol is a full duplex protocol. During most of the protocol dialog a high level synchronization is in effect. For example: the high level packet START OF DOCUMENT requires a high level response of OK or NO before any more high level packets will be sent. But in general packets may be transmitted and received simultaneously.

In most cases simultaneous reception and transmission requires no special processing. But when the low level receives a NAK while sending a packet, it terminates that packet with RESET (octal 177), then RETURN. RESET has no other meaning and is outside the range of characters which encode DX data. It therefore ensures that the checksum received will be faulty, and conserves use of processor and transmission channel, since the packet in progress will have to be retransmitted anyway.

5.2.6 Low Level DX Versus Low Level AX

The difference between low level DX and low level AX protocols is in the startup and shutdown dialog. A DX station will send an INIT packet every five seconds until it receives a response of INIT-ACK or INIT. An AX station will wait for an INIT packet and will respond with an INIT.

Once initialization is complete the operation of DX and AX at the low level is the same. This includes sending and receiving high level data, responding to incoming high level data with ACK or NAK, responding to ACK or NAK appropriately, and all timing parameters.

There is another difference between AX and DX on cancellation of low level connection either in response to an incoming QUIT packet or a high level directive. AX returns to the INITIALIZING state and will respond to an incoming INIT packet with INIT-ACK. DX enters the STOPPED state, and requires operator intervention before it will begin the startup dialog.

5.3 HIGH LEVEL DX

An implementation of the high level of the DX protocol must have the ability to:

1. Respond to user commands.
2. Control system storage.
3. Detect the state of the low level connection (initializing, operating, stopped).
4. Direct the low level to initialize a data connection.
5. Direct the low level to cancel a data connection.
6. Direct the low level to send high level data packets.
7. Respond to high level packets received through the low level.
8. Keep times of the last (high level) packets sent and received.
9. Differentiate between a remote DX station and a remote AX station.

See Appendix G for state transition diagrams for high level DX and AX. These diagrams depict high level protocol states in terms of the packets that may be transmitted and received by each state, and new state resulting from each transmission or reception.

5.3.1 Initialization

Once the low level is initialized, the high level packet IDENTIFICATION is sent. The IDENTIFICATION packet includes a descriptive text message and a code to indicate whether the station is a DX or an AX station. The text message says "You are connected to a <product/version> in <DX or AX> mode." A DX station can initiate and control document transfer, either sending or receiving. An AX station can either receive or send, but does not initiate or control the transfer.

When two DX stations are connected together the OPTIONS packet is sent after the IDENTIFICATION packet. This completes high level initialization.

When a DX station is connected to an AX station, the DX station does not send an OPTIONS packet. Since the AX station has no control of the session it has no choice of options. The AX station can send an optional MESSAGE packet for further identification of itself to the DX user. It can then send any number of PROMPT WITH ECHO or PROMPT WITHOUT ECHO packets to the DX station. Either of these packets requires an ANSWER TO PROMPT packet in return. This exchange allows the AX station to request and receive user validation information such as name and password. The AX station must then send the OPTIONS packet, and high level initialization is complete.

Two AX stations connected together will never initialize at the low level, so high level initialization will never occur.

5.3.2 Operation

The functions that can be performed by the DX or AX station are:

1. To send a message to the remote station.
2. To send a document to the remote station.
3. To receive a document from the remote station.

4. To halt a document transmission in progress.

5.3.2.1 Sending A Message

The text of the message is sent in the MESSAGE packet. Return to the high level operational state. Sending or receiving a MESSAGE packet is allowed at any time during high level operation. Neither sending nor receiving a MESSAGE packet changes the state of the station.

5.3.2.2 Sending a Document

When the local station has a document to transmit, it sends a SEND packet if talking to AX, and a RECEIVE packet if talking to DX. The packet text contains the name of the document to be transmitted. The remote unit can respond in three ways:

1. It can refuse to accept the transfer. A NO packet means that the remote station cannot accept the document. The packet text tells why. Return to the high level operational state.
2. It can ask what to do about a document name conflict. A DOCUMENT OPTIONS packet means that the given name is already in use on the remote station. The packet text specifies what options are available. An ANSWER TO PROMPT packet is required in response to the DOCUMENT OPTIONS making a choice among the options. Only an AX station can send a DOCUMENT OPTIONS packet. Appendix A gives the formats of the packets.
   - If a new name is to be chosen, return to the high level operational state. Receipt of the GOLD-MENU option in the ANSWER TO PROMPT text signals that the dialog is to start over with the local station sending a SEND packet.
   - Otherwise the remote station should return a packet indicating agreement on the name. The ANSWER TO PROMPT text indicates the user's choice that new text is to be added to the top or bottom of the original document, or that new text is to overwrite the original document.
3. The remote station can indicate that the document name is acceptable. If the remote station is in AX mode a SEND packet indicates the document name is acceptable. If the remote station is in DX mode a SEND packet indicates that the name is acceptable.

   Once the SEND packet has been returned from the remote station, the local station sends a START OF DOCUMENT packet, containing the approximate size of the document to be transmitted. The remote station can respond in two ways:
   - It can send a NO packet. The text indicates the reason for the negative response. Return to the high level operational state.
   - It can send an OK packet.

   If the OK packet is received the local station has four choices:
   1. To send A TEXT packet containing 64 characters of document text, and remain in this state, sending text.
   2. To send an END OF DOCUMENT packet containing 64 characters or less of document text, and return to the high level operational state.

3. To send a BYE packet halting the transfer in progress. The text of the packet gives the reason.
4. To send a NO packet halting the transfer in progress. The text of the packet gives the reason.

TEXT and END OF DOCUMENT packets do not require any high level response. A high level response of NO or BYE is allowed at any time. Receipt of a high level NO or BYE will cause the transmission to halt and a return to the high level operational state. The text of the NO or BYE packet indicates the reason for the halt. The END OF DOCUMENT packet indicates to the remote station that the transfer is complete. Return to the high level operational state.

5.3.2.3 Receiving a Document

When the local station wants to receive a document from the remote station it sends a RECEIVE packet to AX and a SEND packet to DX. The text of the packet is the name of the document the local station wants to receive. The local unit then expects one of two responses:

1. The remote unit returns a NO packet. The text of the packet indicates the reason for the negative response. Return to the high level operational state.
2. The remote unit indicates agreement on the name. If the remote unit is in AX a RECEIVE packet indicates agreement on the name. If the remote unit is in DX a RECEIVE packet indicates agreement on the name.

Once the SEND or RECEIVE packet has been returned, the document transmission begins. The local unit expects a START OF DOCUMENT packet. This contains the number of blocks the document currently occupies on the sender's storage device. The local unit can respond in two ways:

1. It can send a NO packet, and return to the high level operational state. The text indicates the reason for the negative response.
2. It can send an OK packet.

If the local unit sends an OK packet it expects one of three responses:

1. To receive a TEXT packet containing 64 characters of document text, and remain in this state, receiving text.
2. To receive an END OF DOCUMENT packet containing 64 characters or less of document text, and return to the high level operational state.
3. To receive a BYE response halting the transfer. The text of the packet gives the reason. Return to the high level operational state.

In addition, the local unit can send a BYE or NO packet at any time causing a return to the high level operational state. The text of the packet indicates the reason for the negative response. No high level response is required for the TEXT or END OF DOCUMENT packets received.

5.3.2.4 Halting a Transfer

A DX or AX station may send a BYE packet at any time. This halts the document transmission in progress if any. The text of the packet indicates the reason for the halt. Return to the high level operational state.

5.3.3 Cancellation

The high level connection may be canceled either by the local or by the remote station. The high level connection cannot be canceled separately from the low level connection. If the user requests cancellation the high level directs the low level to cancel the low level connection packet. If the low level connection is canceled by the remote station the local low level will change to the STOPPED state and no more high level activity will take place.

5.3.4 High Level Timing

If the high level is expecting a packet and receives none for 30 seconds, it directs the low level to cancel the data connection. Only in the high level operational state is a packet not expected within 30 seconds, that is, the state in which the allowable packets are SEND, RECEIVE, MESSAGE, and BYE. In the high level operational state if three minutes elapse without line traffic the high level directs the low level to cancel the connection.

5.3.5 High Level DX Versus High Level AX

During initialization the behavior of a DX station differs from that of an AX station. A DX station will never send an OPTIONS packet to an AX station, because an AX station has no control over the session and thus has no options. When two DX stations are communicating it is assumed that the human operators in control of the session will exercise the appropriate options. When a DX station is communicating with an AX station the AX station may send an optional MESSAGE packet to identify itself to the DX user. It may then request a user name and also a password. This sequence allows it to decide what options to allow the DX station to initiate.

During operation, an AX station never initiates a document transfer as a DX station can. Also, when the DX station wants to send a document with a name that is already in use on the AX station the AX station must send a DOCUMENT OPTIONS packet to ask the operator at the DX station how to handle the conflict. When two DX stations are communicating each operator specifies local names and resolves conflicts locally, so the DOCUMENT OPTIONS packet is never needed.

6 VERIFICATION/ENFORCEMENT

Verification of an implementation of the standard would involve successful connection with a Digital office system product, successful reception of a Digital office system document and successful transmission of the document back to the first system. The returned document would then be compared with the original; there should be no differences.

The standard will be enforced (if at all) by market pressure.

APPENDIX A PACKETS

A.1 SUMMARY OF PACKET TYPES

Type Code	ASCII Code	Packet Name	Level	Requires Response?
140	‘	OK	High	
141	a	DOCUMENT OPTIONS	High	ANSWER
142	b	MESSAGE	High	
143	c	BYE	High	
144	d	START OF DOCUMENT	High	OK or NO
145	e	TEXT	High	
146	f	END OF DOCUMENT	High	
147	g	IDENTIFICATION	High	
150	h	(reserved for future expansion)		
151	i	SEND	High	
152	j	RECEIVE	High	
153	k	OPTIONS	High	
154	l	(reserved for future expansion)		
155	m	(reserved for future expansion)		
156	n	(reserved for future expansion)		
157	o	(reserved for future expansion)		
160	p	NAK	Low	
161	q	(reserved for future expansion)		
162	r	(reserved for future expansion)		
163	s	(reserved for future expansion)		
164	t	NO	High	
165	u	(reserved for future expansion)		
166	v	(reserved for future expansion)		
167	w	(reserved for future expansion)		
170	x	ACK	Low	
171	y	QUIT	Low	
172	z	ANSWER TO PROMPT	High	
173	{	INIT-ACK	Low	
174	|	INIT	Low	
175	}	PROMPT WITH ECHO	High	ANSWER
176	~	PROMPT WITHOUT ECHO	High	ANSWER

A.2 PACKET FORMATS

140 (OK): Affirmative response to START OF DOCUMENT.

141 (DOCUMENT OPTIONS): AX sends this packet to DX when DX asks it to receive a document and gives the document a duplicate name. The packet text contains one character which tells DX which options to display to the user:

040 None
041 Top (T)
042 Bottom (B)
043 Top (T) and bottom (B)
044 Overwrite (O)
045 Overwrite (O) and top (T)
046 Overwrite (O) and bottom (B)
047 Overwrite (O), top (T), and bottom (B)

WPS currently uses only codes 043 and 047. DX must respond with an ANSWER TO PROMPT packet whose packet text shows which option the DX user chose:

040 User pressed Gold MENU to select different name
041 OVERWRITE the existing document contents
042 Add new text to the TOP of the document
043 Add new text to the BOTTOM of the document

142 (MESSAGE): The receiver is to display the packet text on its screen.

143 (BYE): Same as MESSAGE but also halts the transfer in progress, if any.

144 (START OF DOCUMENT): The packet text contains the present size (in blocks) of the document on the sender's storage device, followed by the document's 25 printer settings. Each of these 26 numbers is a 12-bit value encoded into two text characters, according to the scheme in Appendix F.

BL	=	number of blocks of 254 characters each
SI	=	print settings initialized 1 = yes, 0 = no
CP	=	number of copies
PM	=	print margin (left margin)
EX	=	extra half lines
TM	=	top margin
BM	=	bottom margin
PS	=	page size
PI	=	pitch
FR	=	page number to start printing FROM
TO	=	page number to print TO
IP	=	number assigned to the Initial Page reserved
AP	=	automatic pagination, 1 = yes, 0 = no
		reserved
SE	=	Stop before Every page, 2 = first page only, 1 = yes, 0 = no
		reserved
DA	=	darkness 1 = extra dark 0 = normal

TW	=	two wheel printing 1 = yes, 0 = no
DD	=	document destination 0 = LQP, 1 = DP, 2 = Host
CM	=	spaces between columns
R1	=	replacement character 1
R2	=	replacement character 2
		reserved
		reserved

The receiver must reply with an OK or NO packet depending on whether it has enough space to store the document on the requested device.

145 (TEXT): The packet text contains 64 characters of document text.

146 (END OF DOCUMENT): Same as TEXT but indicates the last packet in the document. The packet text may contain fewer than 64 characters.

147 (IDENTIFICATION): The first character of the packet text identifies the software: DX sends 040; AX sends 041. The remainder of the packet text is a descriptive identification message, such as "You are connected with a WS100 in DX."

151 (SEND): This packet is sent to request a document be transmitted from the remote unit to the local unit. The packet text contains the name of the document. The remote unit responds with a SEND packet to indicate agreement on the name.

152 (RECEIVE): This packet is sent to inform the remote unit that a document will be transmitted from the local unit. The packet text contains the name of that document. The remote unit responds with a RECEIVE packet to indicate agreement on the name.

153 (OPTIONS): Sent to DX to tell it what major operations it should let its user select. For instance, if DX is talking to AX and the DX user has typed a "send-only" password, AX tells DX not to make the R(eceive) option available. The packet text contains one character:

040 Allow nothing
041 Allow the B(ye) option
042 Allow the B(ye) and M(essage) options
043 Allow B(ye), M(essage), and S(end)
044 Allow B(ye), M(essage), S(end), and R(eceive)

160 (NAK): A low level response indicating that data has been erroneously received. The sequence number is that of the last packet received correctly.

164 (NO): A high level negative response to a START OF DOCUMENT, TEXT or END OF DOCUMENT packet. The text of the packet gives the reason for the negative response.

170 (ACK): A low level response indicating that data has been correctly received. The sequence number is that of a packet received. ACK asserts that the named packet, and any previous ones since the last ACK or NAK, have been received successfully.

171 (QUIT): A low level message sent by a unit whose user has pressed Gold MENU to recall the Main Menu. Ends communications.

172 (ANSWER TO PROMPT): Solicited by the DOCUMENT OPTIONS, PROMPT WITH ECHO, and PROMPT WITHOUT ECHO packets. The packet text represents information the sender's user typed.

173 (INIT-ACK): A positive acknowledgement of INIT, with the same format.

174 (INIT): Sent initially by DX until receipt of INIT-ACK establishes communication. The first character of the packet text is 040 to indicate the DX/AX protocol. The second character is the version number of the protocol in effect at the sending unit. The third character is the number of NULs (octal 000) the sender of the packet wants the receiver to send after the RETURN character in all future packets.

175 (PROMPT WITH ECHO): This packet asks the receiver to display the packet text on its screen, allow keyboard input, and send the response back in an ANSWER TO PROMPT packet.

176 (PROMPT WITHOUT ECHO): Same as PROMPT WITH ECHO, but also requests that the receiver not display the user's response as it is typed. Used when the response contains secret information, such as a password.

APPENDIX B THE FORMAT OF A WPS DOCUMENT AS TRANSMITTED BY DECDX

The following document format description is presented using the Bachus-Naur Form (BNF) style of notation.

```
 1.  <wps document>  ::=  <wps header> <wps ruler> <wps text>
 2.  <wps text>  ::=  ... EMPTY PRODUCTION ...
 3.                 |  <single character> <wps text>
 4.                 |  <mode> <change> <wps text>
 5.                 |  <cmnd> <special> <wps text>
 6.  <mode>  ::=  <left curly bracket>
 7.  <left curly bracket>  ::=  {
 8.  <cmnd>  ::=  <vertical bar>
 9.  <vertical bar>  ::=  |
10.  <special>  ::=  <tab>
11.                |  <end of line>
12.                |  <end of page>
13.                |  <line modified>
14.                |  <enter composite>
15.                |  <exit composite>
16.                |  <start ruler>
17.                |  <end ruler>
18.  <change>  ::=  <enter boldface>
19.                |  <exit boldface>
20.                |  <enter underline>
21.                |  <exit underline>
22.                |  <enter superscript>
23.                |  <exit superscript>
24.                |  <enter subscript>
25.                |  <exit subscript>
26.                |  <enter auxiliary>
27.                |  <exit auxiliary>
28.  <tab>  ::=  I
29.  <end of line>  ::=  J
30.  <end of page>  ::=  L
31.  <line modified>  ::=  G
32.  <start ruler>  ::=  N
33.  <end ruler>  ::=  O
34.  <enter composite>  ::=  H
35.  <exit composite>  ::=  M
36.  <enter boldface>  ::=
37.  <exit boldface>  ::=  "
38.  <enter underline>  ::=  %
39.  <exit underline>  ::=  $
40.  <enter superscript>  ::=  )
41.  <exit superscript>  ::=  (
42.  <enter subscript>  ::=  +
43.  <exit subscript>  ::=  *
44.  <enter auxiliary>  ::=  '
45.  <exit auxiliary>  ::=
46.  <wps ruler>  ::= <cmnd> <start ruler> <ruler settings> @
                      <ruler settings> <cmnd> <end ruler>
47.  <ruler settings>  ::=  ... EMPTY PRODUCTION ...
```

```
48.                        |  <column number> <ruler setting>
                              <column number> <ruler setting>
                              <ruler settings>
49.   <column number>  ::=  <digit(s)>
50.   <digit(s)>  ::=  ... EMPTY PRODUCTION ...
51.                  |  <digit> <digit(s)>
52.   <ruler setting>  ::=  <decimal tab stop>
53.                       |  <right justified stop>
54.                       |  <left justified stop>
55.                       |  <single spaced left margin>
56.                       |  <ragged right margin>
57.                       |  <double spaced left margin>
58.                       |  <justified right margin>
59.                       |  <word wrap indent>
60.                       |  <paragraph indent>
61.                       |  <centering point>
62.                       |  <one and a half spaced left margin>
63.                       |  <hyphenation zone>
64.                       |  <half spaced left margin>
65.   <decimal tab stop>  ::=  A
66.   <right justified stop>  ::=  B
67.   <left justified stop>  ::=  C
68.   <single spaced left margin>  ::=  D
69.   <ragged right margin>  ::=  E
70.   <double spaced left margin>  ::=  F
71.   <justified right margin>  ::=  G
72.   <word wrap indent>  ::=  H
73.   <paragraph indent>  ::=  I
74.   <centering point>  ::=  J
75.   <one and a half spaced left margin>  ::=  K
76.   <hyphenation zone>  ::=  L
77.   <half spaced left margin>  ::=  M
78.   <wps header>  ::=  <number of blocks> <document name>
                         <print settings>
79.   <number of blocks>  ::=  <digit> <digit>
80.   <creation information>  ::=  <date>
81.   <edit information>  ::= <date> <version> <time> <lines per page>
                               <last session time> <total time>
82.   <print settings>  ::= <protection> <reserved> <settings?>
                            <copies> <print margin> <extra half lines>
                            <top margin> <bottom margin> <page size>
                            <pitch> <first page> <last page>
                            <initial page number> <reserved>
                            <auto pagination> <reserved>
                            <stop printing> <reserved> <darkness>
                            <two wheels> <destination> <column margin>
                            <replacement character 1>
                            <replacement character 2>
83.   <protection>  ::=  <digit> <digit>
84.   <reserved>  ::=  <digit> <digit>
85.   <settings?>  ::=  <digit> <digit>
86.   <copies>  ::=  <digit> <digit>
87.   <print margin>  ::=  <digit> <digit>
88.   <extra half lines>  ::=  <digit> <digit>
89.   <top margin>  ::=  <digit> <digit>
90.   <bottom margin>  ::=  <digit> <digit>
91.   <page size>  ::=  <digit> <digit>
92.   <pitch>  ::=  <digit> <digit>
93.   <first page>  ::=  <digit> <digit>
94.   <last page>  ::=  <digit> <digit>
95.   <initial page number>  ::=  <digit> <digit>
96.   <auto pagination>  ::=  <digit> <digit>
97.   <stop printing>  ::=  <digit> <digit>
98.   <darkness>  ::=  <digit> <digit>
99.   <two wheels>  ::=  <digit> <digit>
```

```
100.   <destination>  ::=  <digit> <digit>
101.   <column margin>  ::=  <digit> <digit>
102.   <replacement character 1>  ::=  <digit> <digit>
103.   <replacement character 2>  ::=  <digit> <digit>
104.   <date>  ::=  <day> <month> <year minus 1900>
105.   <time>  ::=  <hour> <minute>
106.   <day>  ::=  <digit> <digit>
107.   <month>  ::=  <digit> <digit>
108.   <year minus 1900>  ::=  <digit> <digit>
109.   <hour>  ::=  <digit> <digit>
110.   <minute>  ::=  <digit> <digit>
111.   <last session time>  ::=  <minutes>
112.   <total time>  ::=  <minutes>
113.   <minutes>  ::=  <digit> <digit>
114.   <lines per page>  ::=  <digit> <digit>
115.   <version>  ::=  <digit> <digit>
116.   <digit>  ::=  0
117.              |  1
118.              |  2
119.              |  3
120.              |  4
121.              |  5
122.              |  6
123.              |  7
124.              |  8
125.              |  9
126.   <single character>  ::=  NULL
127.                        |  CR
128.                        |  LF
129.                        |  FF
130.                        |  XON
131.                        |  XOFF
132.                        |  SPACE
133.                        |  <printable ASCII character>
```

TECHNICAL CHARACTERS

The following describes the format for a technical character:

<start dead> 40 (space) 61 (1 for USACII) nnn (graphic code) <END dead>
62 (2 for DEC Supplemental)
63 (3 for DEC Technical)

Character	In Memory	In Document	In DX Format
Start Dead	10	Cmd &	vb H
End Dead	15	Cmd ’	vb M

Cmd=77 (octal) in 6-bit ASCII

vb=vertical bar=174 (octal) ASCII

1 means technical character is in USASCII character set

2 means technical character is in DEC Supplemental character set

3 means technical character is in DEC Technical character set

REQUIRED SPACE

The following describes the format for a required space:

<start dead> (space) (space) <end dead>

APPENDIX C FILE FORMAT FOR LAYERED PRODUCTS

DX documents are stored one document per file. DX files are sequential files containing 7-bit ASCII characters only. Characters must be in the range 40 (octal) to 176 (octal). Formfeed, null, tab, and vertical tab characters, as well as other control characters are prohibited in the text. The placement of line breaks in a DX file is not significant.

DX files must start with a valid DX header, consisting of the block size and 25 print settings as transmitted in the START OF DOCUMENT packet. See Appendix A for a description of the format of this packet. See Appendix E for an explanation of the transmitted representation of numeric values. The document name is contained in the file after the header and before the first ruler. This can be up to 64 characters long, and it may not contain any angle brackets. It must be padded to 64 characters with nulls.

APPENDIX D FILENAME CONVENTIONS

The DECdx index is found in the file INDEX.DX. It is an RMS indexed file of fixed length 92 byte records. It uses 3 keys. The keys and record layout are as follows:

Field	Starts at	Length (characters)	
Document number	0	3	! primary key - unique
Create date	3	6	! secondary key, possibly ambiguous
Date last modified	9	6	! 10000 - YYMM, 100 - DD
Time last modified	15	4	! HHMM
Size in WPS blocks	19	6	! WPS blocks are 254 characters
Number of readers	25	2	! 99 max
Number of writers	27	1	! 1 max
Document name	28	64	! secondary key, possibly ambiguous
Date last modified	9	6	! 10000 - YYMM, 100 - DD
Time last modified	15	4	! HHMM
Size in WPS blocks	19	6	! WPS blocks are 254 characters
Number of readers	25	2	! 99 max
Number of writers	27	1	! 1 max
Document name	28	64	! secondary key, possibly ambiguous

The fields for "number of readers" and "number of writers" refer to readers and writers of the associated document file docnnn.dx. Any number of concurrent readers is allowed, but not while there is a writer. Only one concurrent writer is allowed, but not while there are any readers. Violation of this usage could result in loss of data in the DOCnnn.DX file.

The date was placed in the index by getting it in YYMMDD form, then subtracting YYDD from 10000 and subtracting DD from 100, and placing the altered YYMMDD in the index. Thus to place the proper date on the screen the following must be done: subtract YYMM from 10000, the result will be the original YYMM and subtract DD from 100 which will result in the original DD.

APPENDIX E CONVERSION FROM ASCII TO WPS FORMAT

ASCII Character		WPS Representation
nul	has	no translation
^A	has	no translation
^B	has	no translation
^C	has	no translation
^D	has	no translation
^E	has	no translation
^F	has	no translation
^G	has	no translation
^H	has	no translation
tab	becomes	|I
lf	has	no translation
vt	has	no translation
ff	becomes	|L
cr	becomes	|J
^N	has	no translation
^O	has	no translation
^P	has	no translation
^Q	has	no translation
^R	has	no translation
^S	has	no translation
^T	has	no translation
^U	has	no translation
^V	has	no translation
^W	has	no translation
^X	has	no translation
^Y	has	no translation
^Z	has	no translation
esc	has	no translation
fs	has	no translation
gs	has	no translation
rs	has	no translation
us	has	no translation
sp	becomes	sp
(	becomes	(
)	becomes	)
*	becomes	*
+	becomes	+

ASCII Character		WPS Representation
,	becomes	,
-	becomes	-
.	becomes	.
/	becomes	/
0	becomes	0
1	becomes	1
2	becomes	2
3	becomes	3
4	becomes	4
5	becomes	5
6	becomes	6
7	becomes	7
8	becomes	8
9	becomes	9
:	becomes	:
;	becomes	;
<	becomes	<
=	becomes	=
>	becomes	>
?	becomes	?
@	becomes	@
A	becomes	A
B	becomes	B
C	becomes	C
D	becomes	D
E	becomes	E
F	becomes	F
G	becomes	G
H	becomes	H
I	becomes	I
J	becomes	J
K	becomes	K
L	becomes	L
M	becomes	M
N	becomes	N
O	becomes	O
P	becomes	P
Q	becomes	Q
R	becomes	R
S	becomes	S
T	becomes	T
U	becomes	U
V	becomes	V

ASCII Character		WPS Representation
W	becomes	W
X	becomes	X
Y	becomes	Y
Z	becomes	Z
[	becomes	[
]	becomes	]
←	becomes	←
‘	becomes	‘
a	becomes	a
b	becomes	b
c	becomes	c
d	becomes	d
e	becomes	e
f	becomes	f
g	becomes	g
h	becomes	h
i	becomes	i
j	becomes	j
k	becomes	k
l	becomes	l
m	becomes	m
n	becomes	n
o	becomes	o
p	becomes	p
q	becomes	q
r	becomes	r
s	becomes	s
t	becomes	t
u	becomes	u
v	becomes	v
w	becomes	w
x	becomes	x
y	becomes	y
z	becomes	z
{	becomes	|;
|	becomes	|<
}	becomes	|=
~	becomes	|>

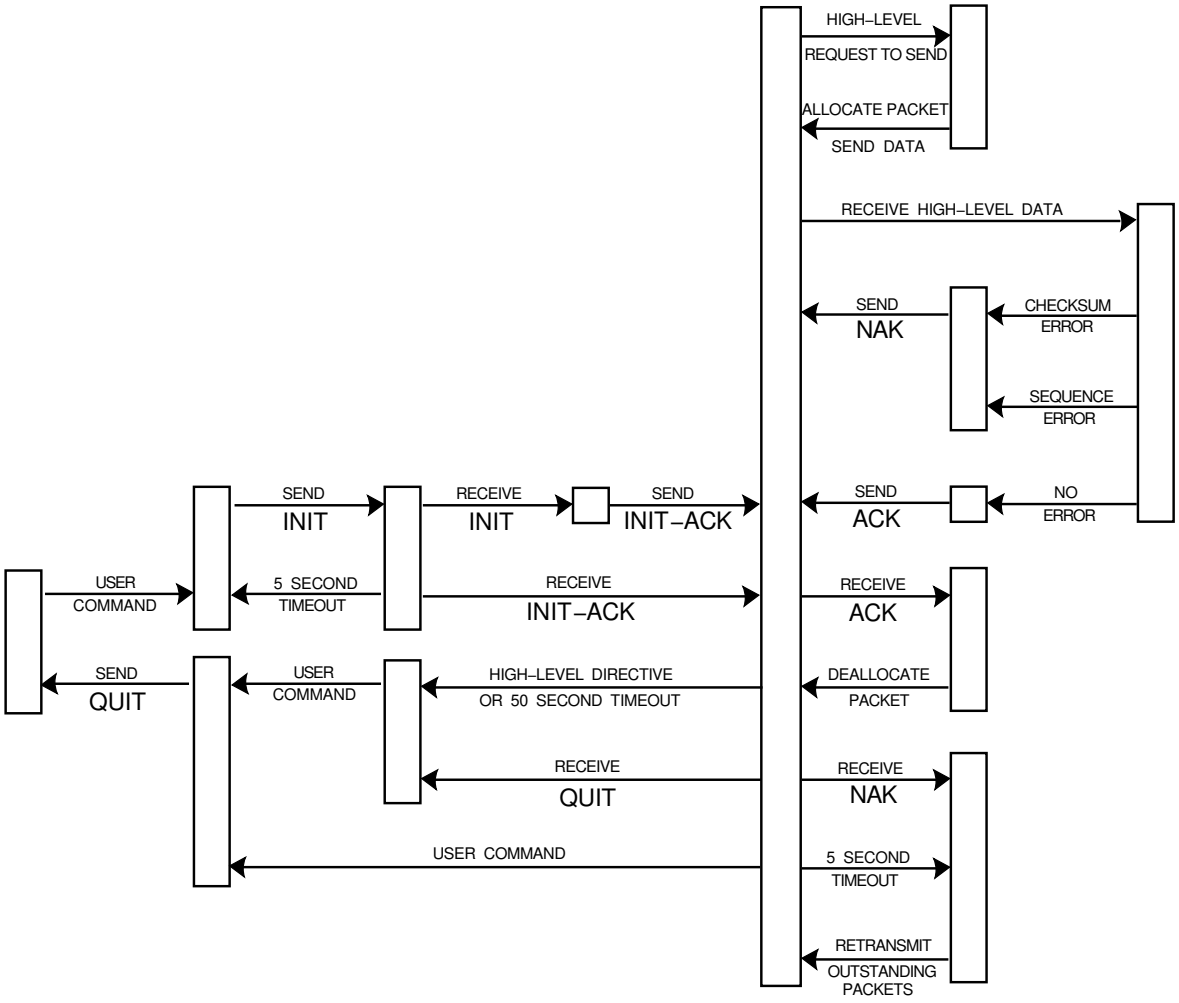
APPENDIX F REPRESENTATION OF NUMERIC VALUES

When transmitting numeric values the DX protocol prohibits transmitting codes which are ASCII control codes (codes from 00 through 37 octal). Therefore an offset of 37 is added to any numeric value transmitted; but 00 is represented by 137. Codes 01 through 37 are transmitted as codes 40 through 76 (octal).

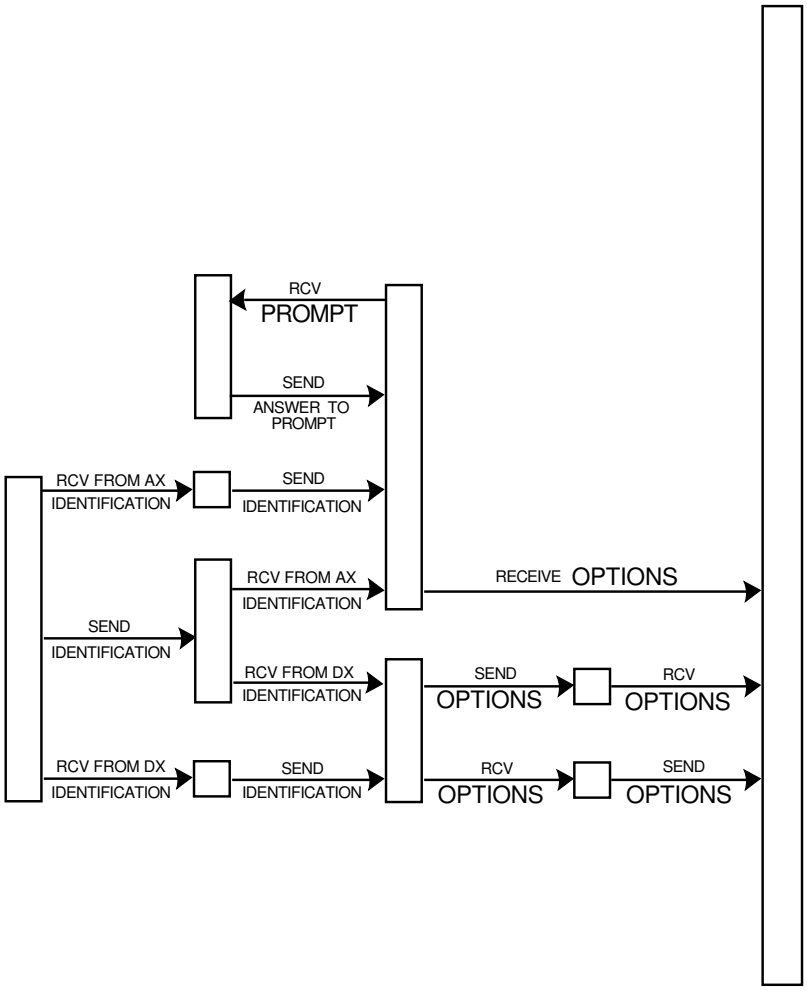
Numeric values in the range of 0000-7777 octal are transmitted as two characters. The high order six bits are transmitted as the first character; and the low order six bits are transmitted as the second character.

APPENDIX G PROTOCOL STATE TRANSITION DIAGRAMS

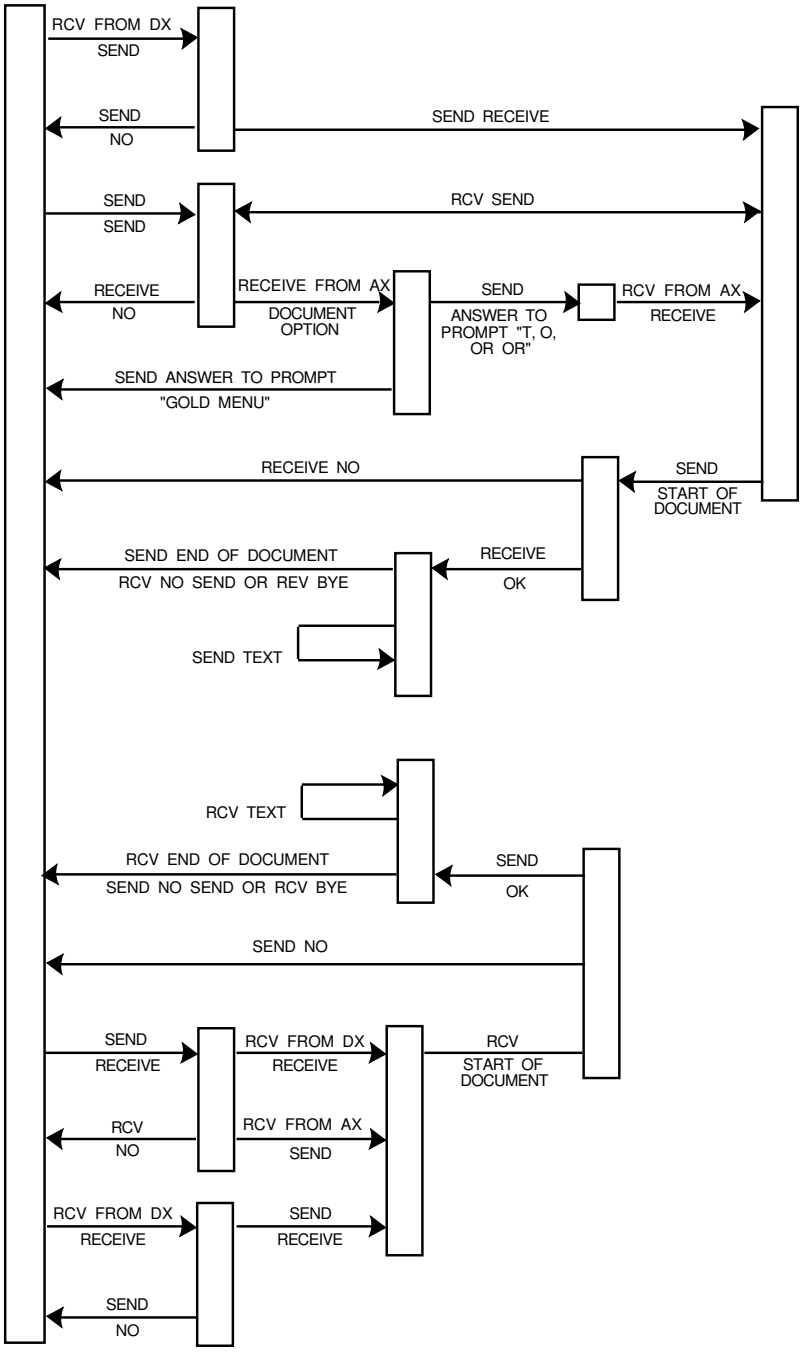
DX LOW LEVEL



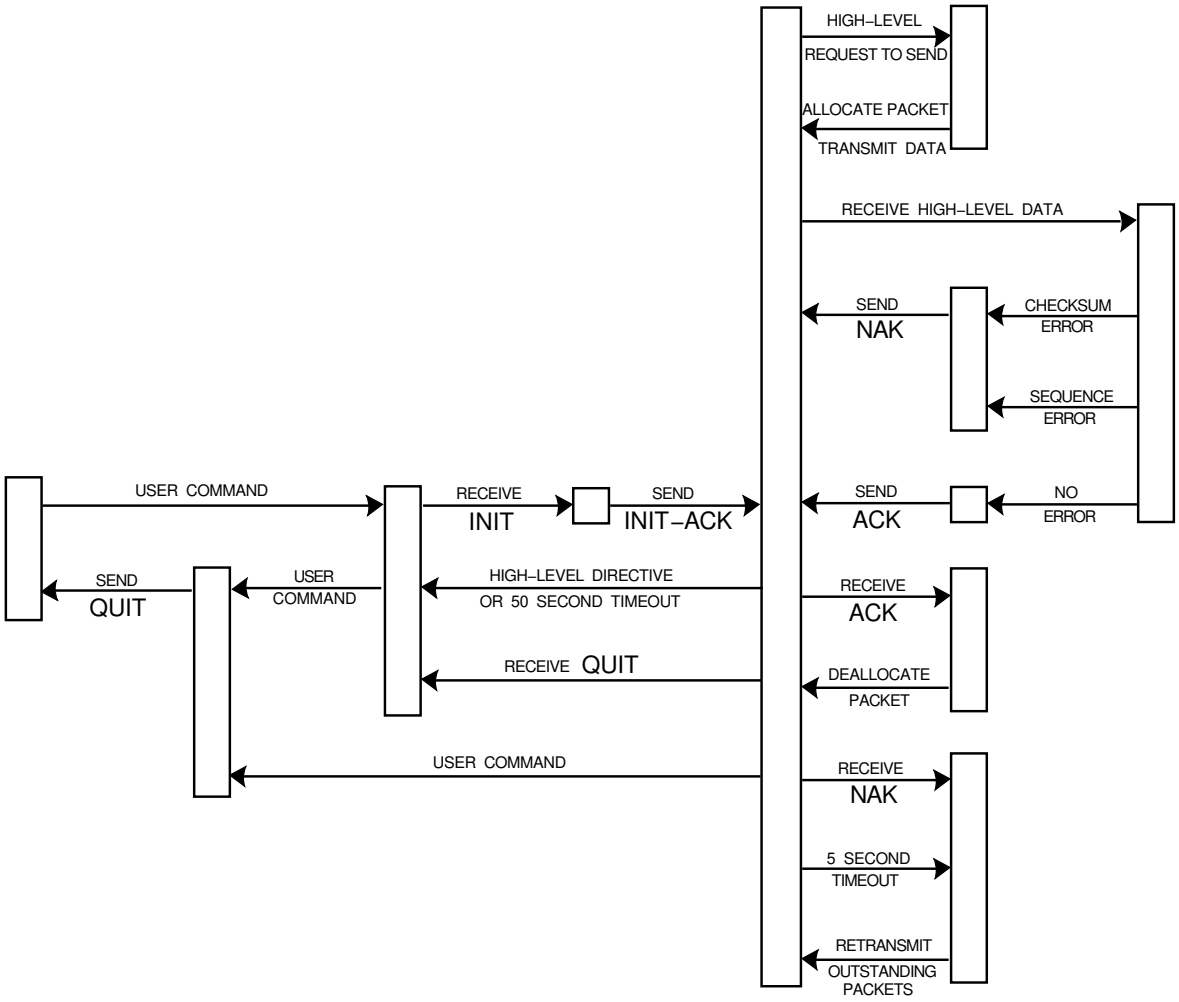
DX HIGH LEVEL INITIALIZATION



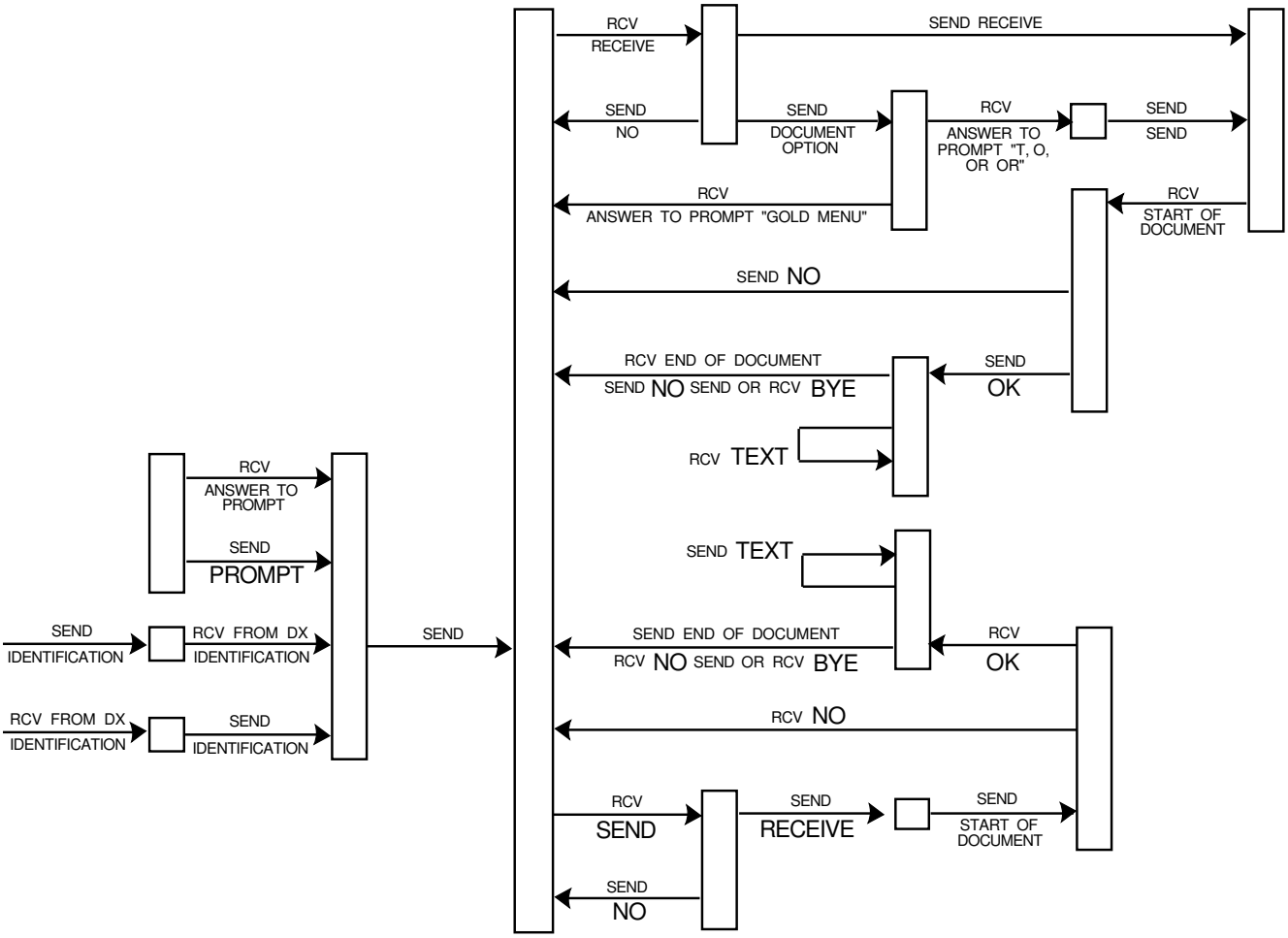
DX HIGH LEVEL OPERATION



AX LOW LEVEL


HIGH–LEVEL
REQUEST TO SEND
ALLOCATE PACKET
TRANSMIT DATA
RECEIVE HIGH–LEVEL DATA
SEND
NAK
CHECKSUM
ERROR
SEQUENCE
ERROR
USER COMMAND
RECEIVE
INIT
SEND
INIT–ACK
SEND
ACK
NO
ERROR
SEND
QUIT
USER
COMMAND
HIGH–LEVEL DIRECTIVE
OR 50 SECOND TIMEOUT
RECEIVE
ACK
RECEIVE QUIT
DEALLOCATE
PACKET
USER COMMAND
RECEIVE
NAK
5 SECOND
TIMEOUT
RETRANSMIT
OUTSTANDING
PACKETS

AX HIGH LEVEL



APPENDIX H DX LIBRARY

This guide is intended for application programmers who already are familiar with VAX/VMS. Readers are assumed to be familiar with the conventions used by the VAX–11 Run-Time Library.

H.1 INTRODUCTION

H.1.1 APPENDIX STRUCTURE

This guide contains five sections.

The introduction discusses the organization and capabilities of the DX Library.

DX Library Routines describes how to use the routines in the library.

User-supplied Routines describes the requirements to user-defined routines that have to interface to the Library routines.

Condition Handling summarizes condition handling within the library.

Callable Library Overview is a graphic representation of the Callable Library.

H.1.2 OVERVIEW

The DX library is a set of callable routines that allow the transfer of WPS documents between VAX/VMS systems and Digital's WS78 word processors and DECmates I, II, III using the DX protocol. This library is part of the DECdx V1.1 software residing on the VAX. In the following, any reference to 'host' implies a VAX/VMS system.

The following types of routines are found in this library:

- DECmate-specific routines
- Document level routines
- Packet level routines
- Easycom routines

These routines gives a user program the capabilities to run either the DX (master) or the AX (slave) protocol for the purposes of transferring WPS documents over a terminal line.

H.1.3 LINKING WITH THE DX LIBRARY

The DX library is called DXLIB.OLB and has to be specifically referred to at link time. For example:

```
LINK/EXE=TEST MAIN,SUB,DXLIB/LIB
```

If the user will make calls to to the AX document level routines, the two objects DXTGET.OBJ and DXTPUT.OBJ in the DECdx kit must also be included at link time.

H.1.4 CALLING LIBRARY ROUTINES

The DX Library conforms to the VAX–11 Procedure Calling Standard and can therefore be called by all native-mode languages under VAX/VMS. However, it should be pointed out that at this stage only the languages Bliss and Koala have been used.

To execute properly, each library procedure requires you to supply parameters. Each parameter must be of the data type and form required by the procedure and must be passed in the proper order by the correct mechanism. You can select your own parameter names, but you must code them as outlined in H.2.

Most routines return a completion value or a function value. Routines called from a high-level language receive this as the value of the function. Routines called in assembly language (MACRO) can access this value in register R0 or R0/R1. The status mnemonics listed in the description of each routine (for example, 'dx$_qui') are defined in a VMS-generated error message module within the DX Library.

For information on the notation used in the parameter descriptions below; see notes in the section 'Summary of Run-Time Library Entry Points' in the VAX–11 Run-Time Library User's Guide.

H.1.5 ORGANIZATION OF THE DX LIBRARY

The library consists of four major parts: DECmate-specific routines, document-level routines, packet-level routines, and Easycom routines.

H.1.5.1 DECmate-Specific Routines

This section consists of two routines—one for sending a document to the DECmate from the host—the other for receiving a document from the DECmate.

Each routine is essentially a set of calls to Easycom- and document-level routines. The routines are written explicitly for use with DECmates that run WPS Easycom software, and provide the capability above the document level routines of switching the DECmate between the CX and AX modes for the purpose of performing automatic document transfer.

NOTE

Document transfer using these DECmate - specific routines is performed under host control since it means use of the DX (master) protocol on the host. WPS software V1.1 or later is a prerequisite.

H.1.5.2 Document-Level Routines

When writing an application program to transfer documents the user has a choice between packet level and document level routines. The document level uses packet level routines, so use of the document level will relieve the user of the need to have detailed knowledge of the DX protocol.

Four routines comprise this section; two for the AX protocol and two for the DX protocol.

Any one of these routines ensure correct delivery of a document - if an irrecoverable error occurs, the routine will return with the appropriate status.

Before any one of these routines can be called, the line must be logically connected. This is accomplished by calling the appropriate connect routine at the packet level.

The DX routines in this section can communicate across the line with the equivalent routines for AX only - DX to DX communication is not possible in the current implementation.

Note that AX to AX communication is not logically possible - the protocol is not symmetric.

Note also that use of the DX (master) protocol routines means that the host controls the document transfer. When the AX routines are used, on the other hand, the word processor is the controlling party since it must run the DX (master) protocol. The two cases are in other words reversed.

H.1.5.3 Packet-Level Routines

This section consists of five packet level routines; one connect routine each for AX and DX, Send and Receive Packet, and Disconnect. The latter three apply to both AX and DX.

Send and Receive packet ensure correct delivery/receipt respectively of a packet. If an irrecoverable error occurs, the routine will return with the appropriate status.

Before Send - or Receive packet are called, the line must be connected. This is accomplished by calling the appropriate connect routine (AX or DX).

H.1.5.4 Easycom Routines

This section contains routines to check if the terminal is a DECmate with Easycom support and to switch the DECmate into user-selected modes (CX, AX or DX).

Note that the DECmate must be in CX mode when receiving an Easycom escape sequence.

NOTE

Easycom support at the WPS end (a prerequisite for the use of these VAX-resident Easycom routines) is available only for DECmates II with WPS software V1.1 or later and DECmates III with WPS software V1.6 or later.

H.2 DX LIBRARY ROUTINES

H.2.1 DECMATE-SPECIFIC ROUTINES

H.2.1.1 SEND DECMATE DOCUMENT

DX$SD_SEND_DOCUMENT

This routine sends a document from the host to a DECmate.

It uses Easycom to switch the DECmate into AX (automatic transfer) mode; the actual document transfer now takes place under host control; the DECmate is then switched back to CX 'blank screen' mode. Any error encountered will cause the DECmate to be switched back to the CX mode.

Note that it is required that the DECmate is in CX mode when this routine is called.

The major operations performed are listed below:

1. Query the terminal to see if it is a DECmate which supports Easycom. If not, return error.
2. Using Easycom switch the remote system to AX mode
3. Create a logical link between host (DX) and remote (AX)

4. Send the specified document
5. Disconnect the DX logical link, returning remote to CX mode
6. Return to caller

If any error is encountered, this routine will return the remote system to CX mode and return a status code to the caller indicating the reason for failure. The return code will conform to the VMS standard for completion value in CALL interfaces, and so is VMS-specific.

Format

```
status.w.w.r = dx$SD_Send_Document      (remote_device.rt.dd,
                                        identification.rt.dd,
                                        document_name.rt.dd,
                                        document_disk.rt.dd,
                                        tbo.rt.dd,
                                        password.rt.dd,
                                        producer.zemlc.r,
                                        producer_data.mt.dd)
```

Arguments

remote_device	The name of the terminal line to the remote system. For example, SYS$COMMAND; __TTG4:; etc.. Passed by descriptor.
identification	An identification text string that will be sent to the word processor at connect time; For example, 'You are connected to a VAX/VMS in DX mode.' Passed by descriptor.
document_name	The name of the document to send. If this string is not null, the name it contains will override current doc.name contained in the .DX file document header block. Passed by descriptor.
document_disk	Remote disk specification. A null string causes the remote to use default ('1'). Passed by descriptor.
tbo	A string to be sent to the remote if there is a conflict in document names. In this case, the remote will use the contents of the string to decide further action: 'T' means add to top of the existing document. 'B' means add to bottom of the existing document 'O' means overwrite the existing document. This parameter is passed by descriptor.
password	DECmate AX password. A nullstring means no password. Passed by descriptor.
producer	Action routine which supplies text to be transmitted. Passed by reference.
producer_data	A parameter for the action routine. Passed by reference.

Completion Codes

dx$_norm	send document completed
dx$_noezcom	remote does NOT support Easycom
dx$_nocntc	could not establish connection
dx$_cntclost	contact lost after retries during document transmission
dx$_qui	remote system explicitly severed communications connection (user hit GOLD-MENU)
rms$_???	any error returned from the producer action routine will be propagated upwards.

H.2.1.2 RECEIVE DECMATE DOCUMENT

DX$RD_RECEIVE_DOCUMENT

This routine allows the host to receive a document from a DECmate. It uses Easycom to switch the DECmate into AX (automatic transfer) mode; the actual document transfer now takes place under host control; the DECmate is then switched back to CX 'blank screen' mode. Any error encountered will cause the DECmate to be switched back to the CX mode.

Note that it is required that the DECmate is in CX when this routine is called.

The major operations involved are listed below:

1. Query the terminal to see if it is a DECmate which supports Easycom. If not, return error.
2. Using Easycom switch the remote system to AX mode
3. Create a logical link between host (DX) and remote (AX)
4. Receive the specified document
5. Disconnect the DX logical link, returning remote to CX mode
6. Return to caller

If any error is encountered, this routine will return the remote system to CX mode and return a status code to the caller indicating the reason for failure.

The return code will conform to the VMS standard for completion value in CALL interfaces, and so is VMS-specific.

Format

```
status.ww.r = dx$RD_Receive_Document    (remote_device.rt.dd,
                                         identification.rt.dd,
                                         document_name.rt.dd,
                                         document_disk.rt.dd,
                                         password.rt.dd,
                                         consumer.zemlc.r,
                                         consumer_data.mt.dd)
```

Arguments

remote_device	The name of the terminal line to the remote system, For example, 'SYS$COMMAND'; '__TTG4:'; and so forth. Passed by descriptor.
identification	An identification string that will be sent to the word processor at connect time, For example,'You are connected to a VAX/VMS in DX mode.' Passed by descriptor.
document_name	The name of the document to receive. Passed by descriptor.
document_disk	Remote disk specification. A null string causes the word processor to use default ('1'). Passed by descriptor.
password	DECmate AX password. A nullstring means no password. Passed by descriptor.
consumer	An action routine to dispose of text received via transmission. Passed by reference.
consumer_data	A parameter for the action routine. Passed by reference.

Completion Codes

dx$_norm	Receive document completed
dx$_noezcom	Remote does NOT support Easycom
dx$_nocntc	Could not establish connection
dx$_cntclost	Contact lost after retries during document transmission
dx$_qui	Remote system explicitly severed communications connection (user hit GOLD-MENU)
dx$_no_such_doc	The requested document does not exist on the remote system
dx$_print_settings	The document's print settings are illegal
dx$_invalid_name	The name of the document to be sent is not legal
rms$_???	Any error returned from the consumer action routine will be propagated upwards.

H.2.2 DOCUMENT-LEVEL ROUTINES

H.2.2.1 AX SEND DOCUMENT

DX$XSA_SDO_AX

This routine transmits a WPS document across a designated line from the host to a DEC word processor. It uses the AX protocol - the word processor must run the DX protocol. Prior to calling this routine, the user must have called the AX connect routine dx$XCA_Con_AX successfully, and the document must be open for read. Closing of the document is also the responsibility of the caller.

When the desired documents have been transferred, the disconnect routine dx$XD_Dis must be called.

Format

```
status.ww.r = dx$XSA_Sdo_AX     (dxcb.wa.r,
                                 pdh_rtn.ra.r,
                                 pdh_arg.wa.r,
                                 packet.wa.r,
                                 producer.ra.r,
                                 producer_arg.ra.r);
```

Arguments

dxcb	The address of the DX control block, passed by reference. (The user must allocate 17 contiguous longwords for this data structure).
pdh_rtn	The address of the parse-document-header routine. Passed by reference.
pdh_arg	The address of an argument for the parse-doc-header routine. Passed by reference.
packet	A character string allocated by the caller to buffer received packet(s). Passed by reference (not descriptor - note).
Producer	The address, passed by reference, of the producer routine.
producer_arg	The address of an argument for the producer routine. Passed by reference.

Completion Codes

dx$_norm	The document was sucessfully transmitted
dx$_bdchar	A text character was outside WPS range.
dx$_qui	Remote system explicitly severed the connection.
dx$_cntclost	Contact lost with remote, after retries.
dx$_bdpak	A bad packet was received.
dx$_wpsmenabo	A 'NO' or a 'BYE' packet was received (WPS aborted transfer).
dx$k_ERD	Error reading the document.

-Packet level completion codes (See 2.3)-

-Completion codes from the parse-document-header routine (See 3.2.2)-

H.2.2.2 AX RECEIVE DOCUMENT

DX$XRA_RDO_AX

This routine receives a document from a DEC word processor. It is unique to WPS documents and uses the AX protocol - the word processor must use the DX protocol.

Prior to calling this routine, the user must have established a protocol connection by calling the AX connect routine dx$XCA_Con_AX successfully, and the document must be open for write.

The closing of the document is also the responsibility of the caller.

When the desired documents have been transferred, a the disconnect routine dx$XD_Dis must be called.

Format

```
status.ww.r = dx$XRA_Rdo_AX     (dxcb.wa.r,
                                 cdh_rtn.ra.r,
                                 cdh_arg.wa.r,
                                 packet.wa.r,
                                 consumer.ra.r,
                                 consumer_arg.ra.r)
```

Arguments

dxcb	The address, passed by reference, of the DX control block (17 contiguous longwords; user-space).
cdh_rtn	The address of a parse-doc-header routine. Passed by reference.
cdh_arg	The address of an argument for the parse-doc-header routine. Passed by reference.
packet	Address of the packet, passed by reference.
consumer	The address of the consumer routine. Passed by reference.
consumer_arg	The address of an argument for the consumer routine. Passed by reference.

Completion Codes

dx$_norm	The document was received successfully.
dx$_qui	The remote system explicitly severed the connection.
dx$_cntclost	Contact lost with remote, after retries.
dx$_wpsmenabo	A 'NO' or a 'BYE' packet received (WPS aborted transfer).
dx$k_EWD	Error writing document.

-Packet level completion codes (See 2.3)-

H.2.2.3 DX SEND DOCUMENT

DX$XSD_DX

This routine transmits a document from the host to a DEC word processor.

It is unique to WPS documents, and uses the DX protocol. The word processor must run the AX protocol.

dx$XSD_DX implements the DX document-level protocol for WPS documents. It calls the packet-level routines to send the text provided by the producer routine.

Prior to calling this routine, the user must have established a protocol connection with the word processor by a successful call to the DX connect routine dx$XCD_Con_DX. The document must be open for read.

Closing of the document is also the responsibility of the caller.

Similarly, after the desired documents have been transferred, the disconnect routine must be called.

Format

```
status.ww.r = dx$xsd_dx          ( dxcb.wa.r,
                                   vfy_doc_name.rt.dd,
                                   new_doc_name.rt.dd,
                                   document_disk.rt.dd,
                                   tbo_routine.ra.r,
                                   packet.wt.r, producer.ra.r,
                                   producer_arg.ma.r ) ;
```

Arguments

dxcb	The address of the DX control block. Passed by reference. (17 contiguous longwords in user-space).
vfy_doc_name	A document name used to verify the document name in the header of the current document. 'vfy_doc_name' and the document name in the header must match. If there is no match, the routine will return with the status 'dx$_nover'. If a nullstring is supplied, this parameter is ignored. Passed by descriptor.
new_doc_name	This name will override current document name contained in the .DX file's document header block. Passed by descriptor.
document_disk	Remote destination disk number. Passed by descriptor.

NOTE

The default disk is '1'. If this parameter has a null value, it will be ignored; otherwise the disk number will be appended to the document name and in effect cause only the 62 first characters of the document name (a WPS document name can be 64 characters long.) to be sent. Be therefore careful with the use of maximally long document names if this parameter is utilized.

tbo_routine	A user-supplied routine that, when called, provides instructions to the remote system in the case when a document already exists on the receiving system with the same name as the one to be sent. The value of the routine itself must be one of the following: '3' - means add to top of the existing document. '4' - means add to bottom of the existing document. '5' - means overwrite the existing document. This parameter is passed by reference.
packet	A character string allocated by the caller to buffer received packet. Passed by reference.
Producer	An the action routine which produces input records. Passed by reference.
producer_arg	A parameter for the producer routine. Passed by reference.

It is assumed that the first call to the producer routine gives the document header which is then supplied to the parse document header routine.

Completion Codes

dx$_norm	The document was sucessfully transmitted
dx$_bdchar	a text character was outside WPS range.
dx$_qui	Remote system explicitly severed the connection.
dx$_cntclost	Contact lost with remote, after retries.
dx$_bdpak	A bad packet was received.
dx$_wpsmenabo	A 'NO' or a 'BYE' packet received (WPS aborted transfer).
dx$_nover	Document name verify error.
dx$_NT_WPS	Error detected on DECmate, but no more information.
dx$_CT_WPS	Error detected on DECmate. Cannot transfer document.
dx$_IWPS	Error detected on DECmate. Incorrect password.
dx$_IU_WPS	Error detected on DECmate. Illegal user.
dx$_YI_WPS	Error detected on DECmate. Your identification message.
dx$_TWPS	Error detected on DECmate. The password.
dx$_NM_WPS	The DECmate has detected a document transfer error.
dx$k_ERD	Error reading the document.

-Packet level completion codes (See 2.3)-

-Completion codes from the parse document header routine (See 3.2.1)-

H.2.2.4 DX RECEIVE DOCUMENT

DX$XRD_DX

This routine receives a document from a DEC word processor. It is unique to WPS documents, and runs the DX protocol. The word processor must run the AX protocol.

dx$XRD_DX implements the DX document-level protocol for WPS documents. It calls the packet-level routines to receive the text, and disposes of it through the consumer routine.

Prior to calling this routine, the user must make a successful call tothe DX connect routine dx$XCD_Con_DX, and have opened the document for write. Closing of the document is also the responsibility of the caller.

Similarly, after the desired documents have been transferred, the disconnect routine dx$XD_Dis must be called.

Format

```
status.ww.r = dx$xrd_dx          (dxcb.wa.r,
                                  packet.wa.r,
                                  consumer.ra.r,
                                  consumer_arg.ma.r,
                                  cdh_document_name.rt.dd,
                                  document_disk.rt.dd)
```

Arguments

dxcb	The address, passed by reference, of the DX control block. (17 contiguous longwords in user-space).
packet	Address of a user-allocated buffer for the packet. Passed by reference.
consumer	Address of an action routine which produces input records. Passed by reference.
consumer_arg	Parameter for the consumer routine. Passed by reference.
cdh_document_ name	The name of the document (sent from the remote to use in constructing the document header.) Passed by descriptor.
document_disk	Disk where requested document resides. If a nullstring is supplied, this parameter is ignored. See notes for this parameter in 2.2.3 above. Passed by descriptor.

Completion Codes

dx$_norm	The document was received successfully.
dx$_qui	The remote system explicitly severed the connection.
dx$_cntclost	contact lost with remote, after retries.
dx$_wpsmenabo	A 'NO' or a 'BYE' packet was received (WPS aborted transfer).
dx$_NT_WPS	Error detected on DECmate, but no more information.
dx$_CT_WPS	Error detected on DECmate. Cannot transfer document.
dx$_IWPS	Error detected on DECmate. Incorrect password.
dx$_IU_WPS	Error detected on DECmate. Illegal user.
dx$_YI_WPS	Error detected on DECmate. Your identification message.
dx$_TWPS	Error detected on DECmate. The password.
dx$_NM_WPS	The DECmate has detected a document transfer error.
dx$k_EWD	- From char. write routine when an error writing document.

-Packet level completion codes (See 2.3)-

-Completion codes from the construct-document-header routine (See 3.2.2)

-Completion codes from the consumer routine (See 3.1.2)-

H.2.3 PACKET-LEVEL ROUTINES

H.2.3.1 AX CONNECT

DX$XCA_CON_AX

This routine initializes a protocol connection between this end of the line running AX and the other end running DX. The remote station (with which a connection is to be made) must run the DX protocol.

Format

```
status.ww.r = dx$XCA_Con_AX     (device.rt.dd,
                                 dxcb.wa.r,
                                 id.rt.dd,
                                 mode.rw.r,
                                 error.ww.r)
```

Arguments

device	The device name (for example '__TTC4:') of an asynchronous, full-duplex serial terminal line to be used for transmission. Passed by descriptor.
dxcb	The address, passed by reference, of the DX control block, for use by DX in maintaining the state of the DX connection. (17 contiguous longwords; user-space).
id	An identification text string that will be sent to the word processor at connect time. Passed by descriptor.
mode	The value, passed by reference, of the mode. 0 to operate as a DX station. 1 to operate as an AX station. - This parameter is currently ignored -
init	A flag which is set if an INIT packet already has been received, for example, in the case of 'DX' being selected from the WPS main menu. Passed by reference.
error	The I/O error propagated from the device I/O. This is a VMS status (for example, 'ss$_timeout'). Passed by reference.

Completion Codes

dx$_norm	Successful connect.
dx$_qui	remote system refused to connect.
dx$_nopl	Internal DX packetpool exhausted.
dx$_cntclost	Contact lost with remote, after retries.

H.2.3.2 DX CONNECT

DX$XCD_CON_DX

This routine initializes a protocol connection between this end of the line running DX and the other end (running AX).

Format

```
status.ww.r = dx$XCD_Con_DX     (device.rt.dd,
                                 dxcb.wa.r,
                                 id.rt.dd,
                                 prompt_arg.wa.r,
                                 error.ww.r)
```

Arguments

device	The device name (for example, '__TTC4:') of an asynchronous, full duplex serial line to be used for transmission. Passed by descriptor.
dxcb	The address, passed by reference, of the DX control block, for use by DX in maintaining the state of the DX connection. (17 contiguous longwords; user-space).
id	An identification text string that will be sent to the remote at connect time; for example, 'You are connected to a VAX/VMS in DX mode.' Passed by descriptor.
prompt_arg	The AX password to be sent to the word processor. A nullstring passed means no password. Passed by descriptor.
error	The I/O error propagated from the device I/O. This is a VMS status (for example, 'ss$_timeout'). Passed by descriptor.

Completion Codes

dx$_norm	Successful connect.
dx$_qui	remote system refused to connect.
dx$_nopl	Internal DX packetpool exhausted.
dx$_cntclost	Contact lost with remote, after retries.

H.2.3.3 SEND PACKET

DX$XS_SENPAK

This routine sends a packet to the remote system according to the DX protocol. Both high- and low-level packets are handled by this routine. The characters in text packets must be within the printable ASCII range.

The routine applies both to the AX and the DX protocols.

A protocol connection must already exist prior to calling this routine.

Format

```
status.ww.r = dx$XS_SenPak      (dxcb.wa.r,
                                 packetcode.rc.r,
                                 textsize.rw.r,
                                 text.rt.r,
                                 error.ww.r)
```

Arguments

dxcb	The address of the DX Control Block. The control block is initialized by the connect routine. (17 contiguous longwords in user-space). Passed by reference.
packetcode	The type of packet to send. Passed by reference.
textsize	Number of characters in the textstring to be included in the packet. Passed by reference.
text	Address of that textstring, passed by reference.
error	The I/O error propagated from lower levels. Passed by reference.

Completion Codes

dx$_norm	The packet was sucessfully transmitted.
dx$_qui	The remote system explicitly severed the connection.
dx$_nopl	Packetpool exhausted.
dx$_cntclost	Lost contact with remote, after retries.

H.2.3.4 RECEIVE PACKET

DX$XR_RECPAK

This routine receives a packet from the remote system. The text characters must have values within the printable ASCII range.

The routine applies to both the AX and the DX protocols.

A protocol connection must already exist prior to calling this routine.

Format

```
packet.wt.r = dx$XR_RecPak      (dxcb.wa.r,
                                 tmo.rw.r,
                                 flags.rw.r,
                                 status.ww.r,
                                 error.ww.r)
```

Arguments

dxcb	The address of the DX Control Block as initialized by the connect routine. (17 contiguous longwords; user-space). Passed by reference.
tmo	Timeout time in seconds Passed by reference.
flags	A bitmask determining types of packets to receive (high-level or low-level or both). Passed by reference.
status	Routine completion code, passed by reference.
error	I/O error propagated from below. Passed by reference.

NOTE

Unlike the other routines, dx$XR_RecPak does not return with its value equal to the status. Completion status is returned in 'status'. The value of the routine is the address of a success ully-received packet or zero if none was received.

Completion Codes

dx$_norm	A packet was received from the remote system
dx$_tmo	No packet was received within the specified time limit
dx$_qui	The remote system explicitly severed the connection
dx$_nopl	The packetpool is exhausted (private pool). The transmission system does not function any more due to unavailability of packetbuffers. The program must be re-compiled with different pool-size parameters.
dx$_ini	The remote system sent an INIT packet.
dx$_bdpak	Too many successive packets received in error.
-some error-	Other error code (low bit cleared) from the ones listed above, that reports failure.

Routine Value

The routine returns the packet address if a high-level packet was received, otherwise '0'.

H.2.3.5 DISCONNECT

DX$XD_DIS

Explicitly severs the protocol connection.

Format

```
status.ww.r = dx$XD_Dis          (dxcb.wa.r,
                                  flag.rw.r)
```

Arguments

dxcb	The address of the control block describing the DX connection (17 contiguous longwords; user-space). Passed by reference.
flag	A flag indicating whether the routine shall send a 'Quit' packet or not; passed by immediate value. = dx$_qui - send 'Quit' = anything else - no packet sent.

Completion Codes

dx$_norm	The DX link has been severed.
-some error-	Any error code (low bit cleared), implying failure.

H.2.4 EASYCOM ROUTINES

H.2.4.1 CHECK IF REMOTE SUPPORTS EZCOM

DX$SENSE

This routine opens a channel on the specified line and checks if the terminal is a DECmate that supports Easycom.

If we're connected to a DECmate with Easycom support then the channel will be left open for subsequent use by other Easycom routines if specified in the argument 'leave_chan_open'. Format

status.wlc.v = dx$sense (leave_chan_open.mw.v)

Arguments

leave_chan_ open	0 = close channel after sensing 1 = leave channel open after sensing Passed by immediate value.

Completion codes

-Success code-	Device sensed successfully
dx$_noezcom	Remote does NOT support Easycom. qio status values VMS-specific I/O status (for example, 'ss$_timeout').

H.2.4.2 SWITCH REMOTE TO AX MODE

DX$SETAX

Switch DECmate into AX mode

Format

status.wlc.v = dx$setax ( )

Arguments

none

Completion codes

-Success code-	Success
dx$_nocntc	Could not establish connection.

Side effects

This routine assumes that a channel has been opened successfully and that it has been established that we are talking to a DECmate with Easycom support by a previous call to dx$sense.

H.2.4.3 SWITCH REMOTE TO CX MODE

DX$SETCX

Switch DECmate into CX mode

Format

```
status.wlc.v = dx$setcx                    ()
```

Arguments

none

Completion codes

-Success code-	Success
dx$_nocntc	Could not establish connection.

H.2.4.4 SWITCH REMOTE TO DX MODE

DX$SETDX

Switch DECmate into DX mode

Format

```
status.wlc.v = dx$setdx                    ()
```

Arguments

none

Completion codes

-Success code-	Success
dx$_nocntc	Could not establish connection.

Side effects

This routine assumes that a channel has been opened successfully and we've established that we are talking to a DECmate by a previous call to routine dx$sense.

H.2.4.5 RETURN REMOTE TO CX MODE

DX$RETURN_TO_CX

Return DECmate into CX mode at the completion of an AX or DX session. This will cause the AX/DX screen to clear pre returnong to CX mode.

Format

```
status.wlc.v = dx$return_to_cx             ()
```

Arguments

none

Completion codes

dx$_norm	Operation was completed. -Packet level completion codes (See 2.3)-

Side effects

This routine assumes that a channel has been opened successfully and we've established that we are talking to a DECmate by a previous call to routine dx$sense.

H.2.4.6 CLOSE AN EZCOM CHANNEL

DX$CLOSE

Deassigns the channel allocated by dx$sense.

Format

```
status.wlc.v = dx$close                    ()
```

Arguments

none

Completion codes

$DASSGN	return status codes (See VAX/VMS System Services Reference Man.)

H.3 USER-SUPPLIED ROUTINES

H.3.1 DOCUMENT-LEVEL INPUT/OUTPUT ROUTINES

The user must provide two routines; one producer and one consumer routine. These routines are the interface between the application- specific file cabinet and the document level routines in the DX Library.

The following must be adhered to:

H.3.1.1 Producer

This is a record-oriented routine. Each call to this routine passes one line of text to the caller. The calling routine (typically one of the document level routines in the DX Library) is not concerned with where this text comes from, but the text source would typically be a document in the application file cabinet. The first call to this action routine provides the header of the document to be sent (DECdx document header size is 116 characters). Successive calls provide one line of text at a time. (A line in a DECdx document can be up to 64 characters in length.)

The routine must place the text in the buffer described by the string descriptor and update the text length in the descriptor accordingly.

Format

producer	(string_descriptor, producer_arg)

Arguments

string_descriptor	The address of a descriptor for temporary storage for the textstring. This descriptor is already defined in the library.
producer_arg	An argument for the producer routine - passed by reference. If no argument is needed, the user must pass a dummy parameter.

Completion codes

RMS$_success	The record is available through the descriptor
RMS$_EOF	This was the last record in the document.

NOTE

For use with DX document routines only. dx$_END_FILE NOTE: For use with AX document routines only. This mnemonic is not included in the VMS-generated error message module, so it will be unknown to any user program linked to the library. To be able to check this status if returned from a library routine, make a definition equivalent to 'dx$_END_FILE = 72' (decimal)(48 hexadecimal) RMS$_???? Errors. These are all returned to the caller with no action taken upon them.

H.3.1.2 Consumer

This is a record-oriented routine that puts away (for example, by writing to disk) the text received over the line. The Consumer works like the Producer but in the opposite direction; with each call to the Consumer the caller must provide a line of text that the Consumer puts 'somewhere', but the actual destination for this text is of no concern to the caller. It is the Consumer itself that has to know what file cabinet it is writing to. For example, the first call to this action routine puts the header of the document received. Successive calls put one line of document text at a time.

The library routines set up the textstring descriptor prior to calling this routine.

Format

consumer	(string_descriptor, consumer_arg)

Arguments

string_descriptor	The address of a descriptor for temporary storage of the textstring. This descriptor is already defined in the library.
producer_arg	An argument for the consumer routine - passed by reference. If no argument is needed the user must pass a dummy parameter.

Completion codes

RMS$_success	The record was successfully put to its destination.
RMS$_???? Error.	Any error will be returned to the caller with no action having been taken upon it.

H.3.2 DOCUMENT HEADER VERIFICATION ROUTINES

— APPLIES TO AX ONLY —

NOTE

The document level routines for the DX protocol have their own header verification routines that adhere to the DECdx document format.

The user must provide two routines; one to construct a DECdx document header (if the application uses the DECdx document format) when receiving from the word processor and one to check (parse) the DECdx document header when sending to the word processor.

H.3.2.1 Construct-Document-Header Routine

Construct a valid WPS formatted document header from the header elements passed by the caller. This routine returns the valid header (padded with ASCII nulls) to the caller.

Format

construct_document_header	(cdh_arg, printset_descriptor, docname_descriptor, header_descriptor)

Arguments

cdh_arg	An argument for the routine, passed by reference. If this is not needed the user must supply a dummy parameter.
printset_ descriptor	The address of a descriptor passed by the caller. It describes the printsettings from the word processor. The printsettings consist of 26 fields of 2 characters each = 52 characters.
docname_ descriptor	The address of a descriptor passed by the caller, describing the document name to be included in the header. A WPS document name can be up to 64 characters long, including the WPS disk number if that is used
header descriptor	The address of a descriptor defined by the caller for the routine to put the final header in.

Completion codes

-Success code-	Successfull completion - header in output string.
rfx_dhp_bpsh	Bad print setting found while constructing a valid header
rfx_dhp_bdnh	Bad document name found while constructing a valid header.
rfx_dhp_dhlh	Bad passed header length found while constructing a WPS header.

H.3.2.2 Parse-Document-Header Routine

Functional description

NOTE

This logic only applies if using the DECdx document format.

Validate a passed WPS formatted document header and return the individual header elements (that is, print settings and document name) to the caller.

Calling sequence from BLISS

parse_document_header	(pdh_arg, header_descriptor, printset_descriptor, docname_descriptor)

Formal parameters

pdh_arg	An argument for the routine, passed by reference. If it is not needed the caller must provide a dummy parameter.
header_ descriptor	The address of a descriptor in which the caller passes the header string to be parsed. The length of the DECdx document header record is 116 characters.
printset_ descriptor	The address of a descriptor in which the routine returns the printsettings. The WPS document printsettings consist of 26 fields of 2 characters each = 52 characters.
docname_ descriptor	The address of a descriptor that contains the document name the routine has found in the header. See 'construct_document_header' for details.

Completion codes

dx$_norm	Successful completion.
rfx_dhp_bpsp	Bad printsetting found while parsing a WPS header.
rfx_dhp_bdnp	Bad document name found while parsing a WPS header.
rfx_dhp_bhlp	Bad passed header length found while parsing a WPS header.

H.4 CONDITION HANDLING

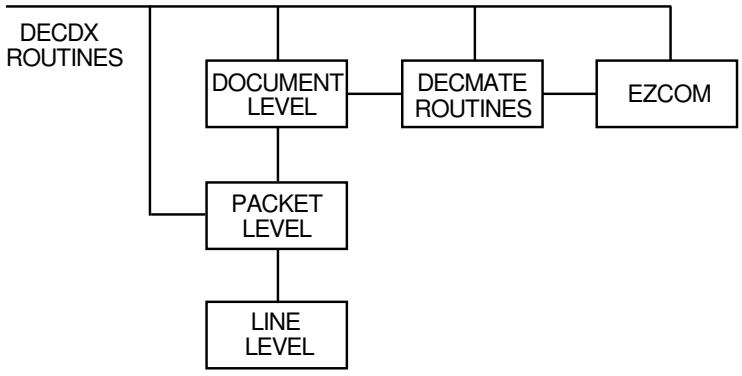
The DX Library condition handling is limited to returning an error status to the calling program. The caller is then free to take whatever action that seems appropriate.

The errors are defined in the module 'dx$mess' in the DX Library. This module was created by the VMS 'Message' utility.

The possible status mnemonics for each routine are listed in section 2 of this appendix.

H.5 CALLABLE LIBRARY OVERVIEW

USER PROGRAM



APPENDIX I REFERENCED DOCUMENTS

I.1 EL-Class Digital Documents

EL-Class Number	Document Title
EL-00052-00	*DEC STD 052-0 Operational Requirements for Serial Terminals and Computer Interfaces Operating as DTEs Connected to EIA RS-232-C or CCITT V.28 Modems*
EL-00111-00	*DEC STD 111-0 DEC Standard for Terminal Synchronization Release*
EL-00138-00	*DEC STD 138-0 Registry of Control Functions for Character Imaging Devices*
EL-00169-00	*DEC STD 169-0 DEC Standard Coded Graphic Character Sets for Hardware and Software*

Use VTX SMC to order copies of EL-class documents from Standards and Methods Control. Send distribution questions to JOKUR::SMC or call DTN: 234-4423.

```
+---------------------------------------------------------------------+
|                          READER COMMENTS                            |
|                                                                     |
| Your  comments  and suggestions  will help  Standards and Methods   |
| Control improve their services and documents.                       |
+---------------------------------------------------------------------+
```

Did you request this document? _____ If so, did it arrive within a satisfactory period of time? _____ Please comment.

What are your impressions of this document? Consider format, organization, completeness, readability, and illustrations.

————————————**FOLD ON THIS LINE**———————————————————————— -

Did you find technical or clerical errors in this document? If so, please specify the page number(s) and the error(s).

Are the instructions for the update package clear? _____
Was an index available? _____ If not, is one needed? _____
Do you have other suggestions for improving this document?

The following information is optional:

Name _____ Mailstop _____

Department _____ Node _____

Send your comments to JOKUR::PROJECTS, or fold, staple, and send this page through interoffice mail to:

```
+------------------------------+
|       READERS' COMMENTS      |
| STANDARDS AND METHODS CONTROL |
|          NRO4/D4             |
+------------------------------+
```